UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL *(If applicable)* 36	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO. / PROJECT NO.	TASK NO. / WORK UNIT ACCESSION NO.
11. TITLE *(Include Security Classification)* SOFTWARE ACQUISITION: EVOLUTION, TOTAL QUALITY MANAGEMENT, AND APPLICATIONS TO THE ARMY TACTICAL MISSILE SYSTEM			
12. PERSONAL AUTHOR(S) Barber, Wayland P.			
13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM ____ TO ____	14. DATE OF REPORT *(Yr., Mo., Day)* 1992, June	15. PAGE COUNT 88
16. SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
17. COSATI CODES (FIELD / GROUP / SUB-GROUP)		18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)* Software Development, Software Acquisition, Total Quality Management (TQM), Army Tactical Missile System (Army TACMS), Multiple Launch Rocket System (MLRS), MICOM Software Engineering Directorate (SED)	
19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*			

Software acquisition has become the critical path in the procurement of Department of Defense (DOD) weapon systems. Software requirements and their complexity have increased at an exponential rate and support requirements now constitute up to 70 percent of the software life cycle costs. This thesis presents the concept of software Total Quality Management (TQM) which focuses on the entire process of software acquisition, as a partial solution to the software acquisition crisis. A software case study, analysis, and lessons learned with applications to the Army Tactical Missile System (TACMS) is presented. A software process control maturity model, a standard software language, and a set of software metrics are presented. A discussion of program manager's responsibilities to implement a process control mechanism to produce quality software products is presented. The principal finding is that software acquisition is the major challenge to a program manager for weapon systems procurement. The major recommendation of this study is that software TQM can be applied to software acquisition.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [XX] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Martin J. McCaffrey		22b. TELEPHONE *(Include Area Code)* (408) 646-2488	22c. OFFICE SYMBOL As/Mf

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603 UNCLASSIFIED

T257838

SOFTWARE ACQUISITION: EVOLUTION, TOTAL QUALITY MANAGEMENT AND APPLICATIONS TO THE ARMY TACTICAL MISSILE SYSTEM

by

Wayland P. Barber
Captain, United States Army
B.S., M.E., Northeastern University, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
June 1992

ABSTRACT

Software acquisition has become the critical path in the procurement of Department of Defense (DOD) weapon systems. Software requirements and their complexity have increased at an exponential rate and support requirements now constitute up to 70 percent of the software life cycle costs. This thesis presents the concept of software Total Quality Management (TQM) which focuses on the entire process of software acquisition, as a partial solution to the software acquisition crisis. A software case study, analysis, and lessons learned with applications to the Army Tactical Missile System (TACMS) is presented. A software process control maturity model, a standard software language, and a set of software metrics are presented. A discussion of program manager's responsibilities to implement a process control mechanism to produce quality software products is presented. The principal finding is that software acquisition is the major challenge to a program manager for weapon systems procurement. The major recommendation of this study is that software TQM can be applied to software acquisition.

Thesis
B21717
c.1

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. GENERAL

Software acquisition has become the long pole in the tent for systems acquisition. Weapon Systems have become extremely complex. As a result, the software development has also become more technically challenging and costly. As systems are upgraded, during upcoming austere budget years, software development will receive more attention. This study of software acquisition, from a program manager's (PM) point of view, will document and use a case study to show current aspects of the software acquisition process as they apply to the present day acquisition life cycle of a weapon system.

B. AREA OF RESEARCH AND OBJECTIVES

The primary focus of this research paper will be on the evolution of the software acquisition and software applications as they apply to the Army Tactical Missile System (TACMS) from a PM's point of view. Additionally, aspects of Total Quality Management (TQM) will be addressed as they apply to software acquisition. In conclusion, lessons learned will be reviewed. In addition, a list of appropriate questions a PM should ask, during the acquisition will be furnished. Answers to these questions will provide a PM information as to the status of the software development.

C. RESEARCH QUESTIONS

1. Primary Research Question

How can the software acquisition process be improved?

2. Subsidiary Questions

- What situations, from a historical point of view, have caused the current software crisis?
- How can Total Quality Management principles be applied to the software acquisition process?
- What lessons can be derived from the Army Tactical Missile System software acquisition process?
- What questions should a PM ask to ensure that the software acquisition process is under control?
- What are the similarities and the differences between software and hardware acquisition?
- What is the quality of a contractor's programming staff?

D. SCOPE

The main thrust of this thesis will focus on the Army Tactical Missile System's software evolution. It begins with the software for the Multiple Launch Rocket System (MLRS) and continues through the ongoing efforts for a potential Block II upgrade. The MLRS System, currently in the Army inventory, is the basic system that the Army TACMS uses as a platform for the Army TACMS missile. With a combination of software and hardware modifications, the MLRS System was upgraded to accommodate the Army TACMS missile. The software modifications to accommodate the Army TACMS will be investigated. Issues pertaining to future modifications will also be presented. Aspects of Total Quality Management (TQM) will be discussed with respect to software acquisition. This thesis will be limited to the historical aspects of the development of the software acquisition as they apply to the Army TACMS software development.

E. METHODOLOGY

The first objective of this study will be to conduct an in-depth historical overview of weapon system software acquisition. This will be accomplished by reviewing applicable publications to include the latest periodicals and recent studies conducted at the Naval Postgraduate School. The next step will be to conduct a Defense Logistics Studies Information Exchange (DLSIE) search for software related information. This initial investigation will provide the necessary information required from an historical perspective and evolve into a case analysis. Interviews of personnel internal and external to the Army TACMS project office will be conducted. Additionally, personnel from the MLRS project office, contractor support offices, and the Missile Command (MICOM) Software Engineering Directorate (SED) will be interviewed.

F. BENEFITS OF STUDY

The primary benefit of this study will be the documentation of the software acquisition for the Army Tactical Missile System. Secondary benefits will focus on the potential uses of TQM for the software acquisition process. A tertiary benefit will be the generation of a generic set of questions PMs should ask about software to ensure that they keep abreast of the status of the software in the acquisition process.

G. ORGANIZATION

Chapter II is an historical perspective which leads up to the current software situation a PM faces. Current regulatory constraints, standards, and background literature are presented.

Chapter III introduces the concept of TQM and possible applications to the acquisition of software.

Chapter IV is an analysis of the Army Tactical Missile System software acquisition process. Additionally, aspects of TQM are presented as they apply to this process.

Chapter V provides an analysis of the Army TACMS software development process and culminates with lessons learned to date.

Chapter VI states the conclusions and recommendations, summarizes the answers to the research questions, and concludes with recommendations for future research.

II. HISTORICAL BACKGROUND

A. INTRODUCTION

General William G.T. Tuttle, Jr., Commanding General, U.S. Army Materiel Command (AMC), opened the Army Executives for Software (ARES) forum, 17 July 1991, by briefing the importance of software as a force multiplier for the U.S. Army. His briefing opened with an explanation that the reason for conducting the forum was "because computer software is about to consume us...if it hasn't already [Ref. 1]." General Tuttle's personal appearance at this forum is indicative of the emphasis the Commanding General, Army Materiel Command, currently places on software acquisition.

General Tuttle proceeded to explain that the entire acquisition community has a stake in software for the following reasons:

- The user depends on technology to defeat the threat.
- Technology depends on software.
- Software enhances warfighting capability.
- Software can generally be changed cheaper/faster than hardware. [Ref. 1]

In culmination, General Tuttle reminded the acquisition community that they should remember the following points:

- Software is not an add-on to our weapon systems--it is an inherent contributor to our ability to fight.

- Our software technology is well ahead of our software management--we need to close the gap.

- The first step - awareness of the issues and our role in closing the gap. [Ref. 1]

General Tuttle sets the stage for the modern day philosophy and the challenges facing the PM and the software acquisition community. This thesis will stress his emphasis on the importance of software acquisition.

B. SOFTWARE HISTORY

Software acquisition is the greatest challenge, and the biggest bottleneck, facing the present day PM when acquiring the extremely complex weapon systems of the modern technological era. Software engineering is the application of scientific and managerial disciplines required to design, produce, modify, and maintain software products. No clearly defined, standardized, software engineering process exists for the acquisition of our weapon systems. Is software "engineering" an art or science? Presently, few quantifiable methods exist to allow for measurement and statistical analysis of an ill-defined software acquisition process. Weapon systems software acquisition is still in its infancy. Education and knowledge of the past can perhaps prove to be the PM's best weapon against the poor software acquisition methodology currently employed. The remainder of this section will be devoted to the basic history of software acquisition and the challenges it presents.

1. Historical Perspective

Mission Critical Computer Resources Management Guide, a publication of the Defense Systems Management College [Ref. 2], provides one of the best capsulized sources illustrating the exponential growth associated with the software industry. The term "software engineering" was not used until 1968 and computer software development is less than 40 years old. The realization that software engineering is still in its infancy as an industry, coupled with the associated learning curves for this profession, indicates why software acquisition is presently the bottleneck of the entire Department of Defense (DOD) acquisition process.

Some of the reasons for the rapid growth in the software industry are evidenced by tremendous advances in hardware and its associated capabilities. Presently microprocessor chip technology and integrated circuits (ICs), the building blocks for computer systems, have been greatly reduced in size, weight, power requirements, and cost.

Most weapon systems have become extremely complex and are software driven. Because it involves no physical change, software is inherently more flexible than hardware. Unfortunately software, unlike hardware, is an intangible entity. It is not readily understood by far too many managers. The above cited examples have been major contributors to the software "crisis" facing PMs.

The *Mission Critical Computer Resources Management Guide* provides some excellent insights to life cycle cost trends, and are presented in Figure 2-1 [Refs. 2, 3]. As a percent of current total acquisition costs, software development and support costs are today approximately 80% (20% hardware), compared to only 20% for software (80% hardware) costs in the mid 1960's era. Another significant cost trend in the software cost explosion is shown in Figure 2-2 [Ref. 2:Fig 7-2]. Of the total software acquisition life cycle requirement, 70% of the cost is associated with sustainment (software support/maintenance). Another interesting historical observation is..."DOD found that the average cost of generating a line of code is about seventy-five dollars while the average cost of modifying a line of code late in the development cycle or after software delivery is four-thousand dollars [Ref. 2:7-7]."

Table 2-1 shows the relative cost to fix software as a function of the software developmental phase. This table indicates where software errors are introduced and found during the software acquisition life cycle. The most significant information is shown for the operations and maintenance phase. Only five percent of the software errors are introduced while 22 percent are found. This phase contributes between ten and 100 times the cost to fix software errors as compared to correcting software errors during the requirements analysis phase. This trend clearly indicates that software support is becoming the bottleneck within the bottleneck of the software acquisition process.

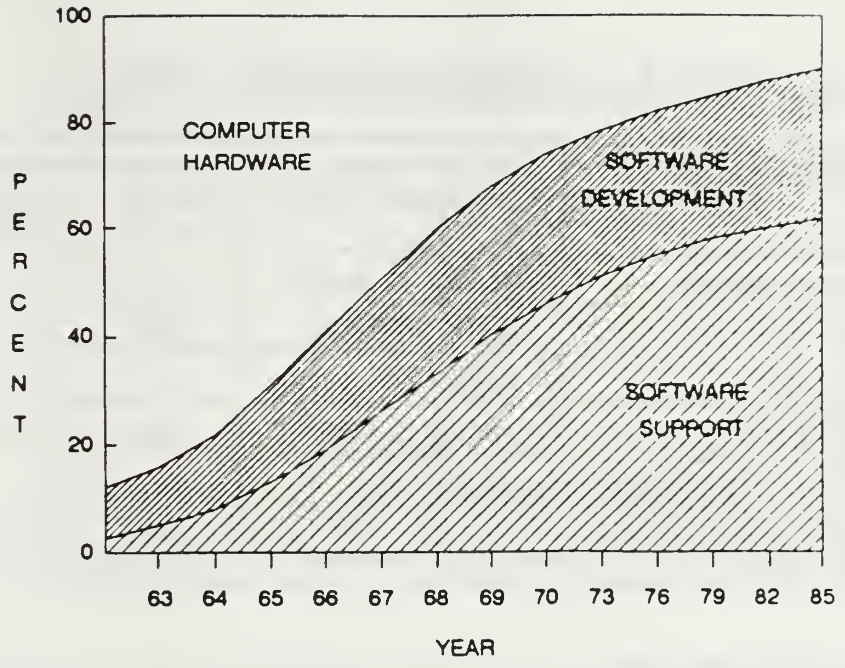

Figure 2-1. Life Cycle Cost Trends

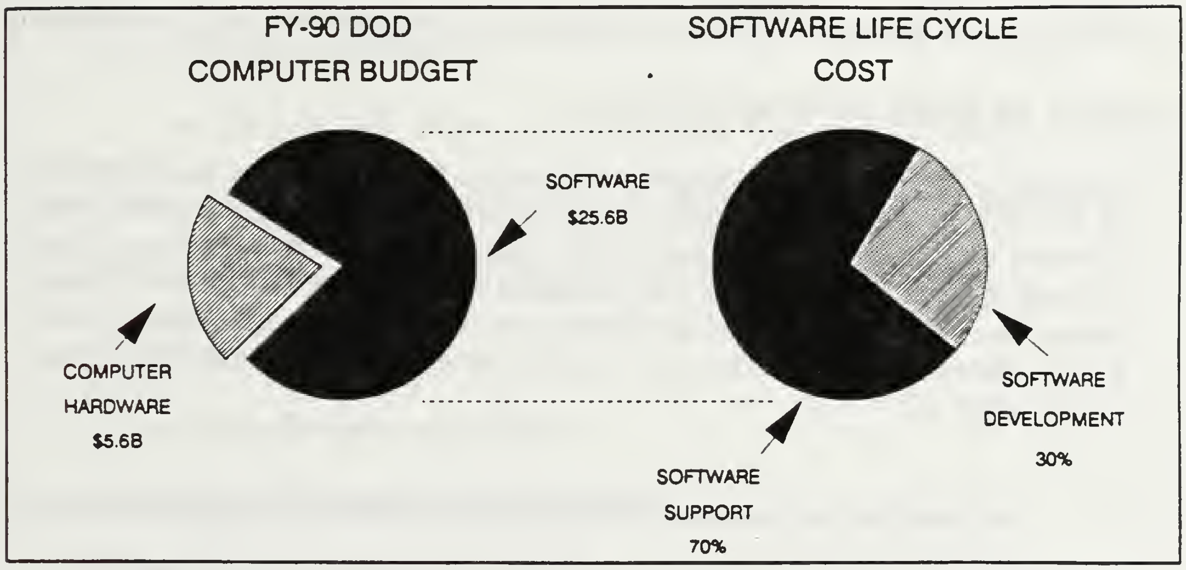

Figure 2-2. Software Costs

TABLE 2-1. COSTS OF SOFTWARE FIXES

SOFTWARE DEVELOPMENT	DEV $	ERRORS INTRODUCED	ERRORS FOUND	RELATIVE COST OF ERRORS
REQUIREMENTS ANALYSIS	5%	55%	18%	1.0
DESIGN	25%	30%	10%	1-1.5
CODE & UNIT TEST	10%			
INTEGRATION & TEST	50%	10%	50%	1.5-5.0
VALIDATION & DOCUMENTATION	10%			
OPERATIONS & MAINTENANCE		5%	22%	10-100

2. The Achilles' Heel

James Kitfield provides some insights to the dilemma facing DOD with respect to software acquisition by stating that the demand for software is increasing by 25% per year while the education system only generates 4% more programmers per year. He further states the following:

> Consider only that while computer performance has increased a thousandfold in the last 30 years, due largely to the incredible shrinking microchip, the software that instructs computers is still crafted by a painstaking process that is much the same as it was in 1960. Imagine that at a time when the United States faces a severe shortfall of civilian and military software professionals, DOD's demand for new software is equal to the entire amount it currently has to use. [Ref. 4]

Kitfield further illustrates the rapid expansion of software and the resultant complexity in our weapon systems by the following comparison [Ref. 4]:

TABLE 2-2. SYSTEM AND SOURCE LINES OF CODE

SYSTEM	SOURCE LINES OF CODE
F-4	Virtually none
F-16D	236,000
C-17	750,000
B-1B	1,200,000
ATF	5,000,000 to 7,000,000
SDI	25,000,000 (est.)

With this situation facing the DOD software acquisition community, it becomes readily apparent why General Bernard Randolf, chief of Air Force Systems Command, described software as an "Achilles' heel" in weapon systems development.

3. Software Risk Items

As part of the Software Operational Test and Evaluation (OT&E) Symposium, 1 November 1990, Barry Bohem, Defense Advanced Research Projects (DARPA), presented the top ten software risk items as follows [Ref. 5]:

- Personnel shortfalls.
- Unrealistic schedules and budgets.
- Developing the wrong software functions.
- Developing the wrong user interface.
- Gold plating.
- Continuing stream of requirements changes.
- Shortfalls in externally-furnished components.

- Shortfalls in externally-performed tasks.
- Real-time performance shortfalls.
- Straining computer science capabilities.

The above information is a further example of the confrontation facing the present day software environment.

4. Snowball Effect

Lieutenant General Jerome B. Hilmes, Director Information Systems for Command, Control, Communications and Computers, describes the software crisis as a snowball effect. The initial problem stems from inadequate requirements definition leading to inadequate development of resources. As the snowball increases in its size, while rolling down the "acquisition hill" at an ever increasing velocity, excessive changes occur during development causing schedule and cost growth. Further down the "acquisition hill" at increasing velocity, the delivery of immature products result, causing software and system failures. When the snowball reaches its final "resting" place, the software crisis is readily apparent by excessive maintenance costs. The result is unfavorable General Accounting Office (GAO), and Inspector General (IG) reports, with congressional inquiries and oversights.

5. Historical Summary

Software acquisition can be described as being in a crisis situation for many of the reasons previously outlined. The crisis becomes apparent when viewed from the PM's point of view, given the rapid expansion of software requirements for weapon systems relative to the availability of a mature software acquisition process. The proceeding illustrations define the present day culture for software acquisition as it applies to the PM within the DOD.

C. PRESENT SITUATION

The Army Acquisition Executive, Stephen K. Conver, recently discussed the success stories from operation Desert Storm [Ref. 6]. The following weapon systems were highlighted as success stories:

- Joint Surveillance and Target Attack Radar System (JSTARS).
- Apache (AH-64) armed with Hellfire missiles.
- Multiple Launch Rocket System (MLRS).
- Army Tactical Missile System (TACMS).
- Bradly Fighting Vehicle Systems (BFVS).
- Abrams (M-1 Tank).

Mr. Conver additionally remarked on the importance of maintaining the "precious technological edge [Ref. 6]." The success stories associated with the Southwest Asia deployment are indicative of the key roll that software plays in modern weapon systems. Each of the previously cited weapons have software intensive subsystems.

Brigadier General William J. Mullen III, Deputy Director of Operations (J3), Forces Command, further emphasized the importance of software in operation Desert Storm [Ref. 7]. His observations, addressed to the Army Executives for Software (ARES) forum on 17 July 1991, cited specific software impacts as a combat multiplier on the modern battlefield. His first example was the Patriot missile system. He noted the Patriot's increased capability to shoot down the SCUD missile was attributable to a major upgrade in only the software. Software was a true force multiplier. Other examples Brigadier General Mullen included in his presentation were:

- Apache (AH64)
- Prisoner of War Information System-2 (PWIS-2)
- TOW
- AN/APR-39 Radar Warning Receiver System

Brigadier General Mullen clearly articulated the importance software played in Southwest Asia by stating: "It is apparent we are not going to do our jobs without software." [Ref. 7]

Both Mr. Conver and Brigadier General Mullen emphasized the important role software plays in weapon systems, the versatility software provides, and how the future depends on quality software.

D. REGULATIONS AND GUIDANCE

This section is devoted to describing the minimum set of reference materials a PM should be well versed in for software and software acquisition.

1. *Mission Critical Computer Resources Management Guide*

The *Mission Critical Computer Resources* (MCCR) *Guide* is a good starting point for a program manager to obtain a broad perspective of software acquisition policies and terminology set forth in the Department of Defense (DOD). This guide gives a brief synopsis and history of computer systems. It then evolves into acquisition policy, to include applicable regulations and policies. Additionally, the following topics are covered: software support, test and evaluation, configuration management, and metrics. The MCCR Guide is a must starting place for all people interested in software development. [Ref. 2]

2. DOD Directive 5000.1

Department of Defense Directive 5000.1, dated February 23, 1991, is the top level document that "establishes a disciplined approach for acquiring systems and materiel that satisfy the operational user's needs [Ref. 8]." This regulation does not specifically address software acquisition. It does provide an excellent road map to all applicable documentation and regulations pertaining to weapon systems and software acquisition.

3. DOD Instruction 5000.2

Department of Defense Instruction 5000.2, dated February 23, 1991, Part 6, Section D, "Computer Resources," supersedes both DOD Directive 5000.29, "Management of Computer Resources in Major Defense Systems," dated April 26, 1976, and DOD Directive 3405.2, "Use of Ada in Weapon Systems," dated March 30,

1987. This directive sets the standards and procedures required for the procurement of embedded computer resources within DOD. Additionally, all applicable statutes, regulations, and standards are listed. [Ref. 9]

4. DOD-STD-1467 (AR)

Department of Defense Standard 1467 (AR), "Military Standard Software Support Environment," dated 18 January, 1985 [Ref. 10], establishes the minimum requirements for the contractor to construct a Developmental Software Support Environment (DSSE), and to ensure the compatibility of this environment with a contracting activity's designated Life Cycle Software Support Environment (LCSSE). The most important element of this document establishes the necessary procedures PMs need to ensure life-cycle software support requirements are properly defined.

5. DOD-STD-2167A

Department of Defense Standard 2167A, "Defense System Software Development," dated 29 February 1988, outlines the requirements for software acquisition with respect to development, sustainment, and life-cycle support of software systems. This standard also addresses the process of tailoring the specification for software and firmware development. This document provides the program manager with at least a minimum set of standards to begin the software acquisition process. [Ref. 11]

6. DOD-STD-2168

Department of Defense Standard 2168, "Defense System Quality Program," dated in 1988, provides current information on how to set up a quality software assurance program. [Ref. 12]

7. Military Handbooks

The following handbooks will provide additional information on software:

- MIL-HDBK-286, "Software Quality Evolution" [Ref. 13]
- MIL-HDBK-287, "A Tailoring Guide for DOD-STD-287, Defense System Software Development." [Ref. 14]
- MIL-HDBK-782, "Software Support Environment Acquisition." [Ref. 15]

8. Army Regulations and Pamphlets

The following are Army documents associated with embedded software:

- AMC-R, 70-16,"Management of Computer Resources in Battlefield Automated Systems" [Ref. 16], establishes the requirements for a PM to ensure software is effectively planned, developed, acquired, tested, fielded, and supported.
- AMC-P, 70-13, "Software Management Indicators, Insight" [Ref. 17], establishes metrics for PM use.
- AMC-P, 70-14, "Software Management Indicators, Quality Insight" [Ref. 18], also establishes metrics for PM use.

These are the minimum set of publications necessary for a PM to review for software acquisition as it applies to the Army.

E. SUMMARY

This historical background chapter began with an opening from the AMC commander, General Tuttle, stating his philosophy on the importance of software and software acquisition as it applies to the modern Army. The next portion was devoted to the historical situation that has produced the current situation that confronts the software industry at the present time. One section was structured to list the more important literature, guidance, and regulations a PM should have a firm grasp on to alleviate the software bottleneck that currently exists within the DOD. The next chapter will introduce the philosophy of Total Quality Management (TQM) as it applies to the process of software acquisition.

III. TOTAL QUALITY MANAGEMENT AND SOFTWARE DEVELOPMENT

A. INTRODUCTION

This chapter is designed to present possible applications of Total Quality Management (TQM) for the process of software acquisition. Dr W. Edwards Deming's philosophy on TQM provides the initial basis for this discussion. His methodology on TQM is then applied to software acquisition. Essential characteristics of TQM, to include process control, standardization, and the use of measuring devices (metrics), as they apply to software acquisition are presented. The necessity of upper management commitment to the customer and continuous process improvement are also addressed.

B. TOTAL QUALITY MANAGEMENT PHILOSOPHY

This section introduces the concept of Total Quality Management (TQM). The basic philosophy of TQM principles from a Department of Defense (DOD) perspective is stated as:

> Total Quality Management (TQM) is both a philosophy and a set of guiding principles that represent the foundation of a continuously improving organization. TQM is the application of quantitative methods and human resources to improve the material and services supplied to an organization, all the processes within an organization, and the degree to which the needs of the customer are met, now and in the future. TQM integrates fundamental management techniques, existing improvement efforts, and technical tools under a disciplined approach focused on continuous improvement. [Ref. 19]

This definition provides a philosophy that all DOD acquisition managers can use during the procurement process for both software and hardware.

The DOD definition of TQM comes directly from the teachings of Dr. W. Edwards Deming, considered the pioneer in this field. His basic belief is that high quality leads to high productivity and customer satisfaction enhances productivity because rework of the product is decreased or eliminated, no scrap is generated, no time is required to correct defects, and the customer remains satisfied. Quality is also the primary requirement for any product produced.

In providing any goods or services emphasis should be placed on both the ultimate customer (user) and each of the intermediary customers engaged in the process. In addition, a clearly defined process to identify the organizational structure and procedures for work activities is necessary. Standardization and statistical process control (SPC) procedures to reduce variations that cause poor quality also need to be defined and implemented.

TQM also requires a feedback loop be identified. This feedback loop provides the necessary information for the process that allows for change in the system. It provides the core requirement for continuous process improvement. This methodology will be used to analyze software acquisition as it applies to TQM.

Dr. Deming has consolidated his beliefs on TQM into the following 14 points:

1. Create constancy of purpose toward improvement of product and service, with the aim to become competitive and to stay in business, and to provide jobs.

2. Adopt the new philosophy. We are in a new economic age. Western management must awaken to the challenge, must learn their responsibilities, and take on leadership for change.

3. Cease dependence on inspection to achieve quality. Eliminate the need for inspection on a mass basis by building quality into the product in the first place.

4. End the practice of awarding business on the basis of price tag. Instead, minimize total cost. Move toward a single supplier for any one item, on a long-term relationship of loyalty and trust.

5. Improve constantly and forever the system of production and service, to improve quality and productivity, and thus constantly decrease costs.

6. Institute training on the job.

7. Institute leadership. The aim of supervision should be to help people and machines and gadgets to do a better job. Supervision of management is in need of overhaul, as well as supervision of production workers.

8. Drive out fear, so that everyone may work effectively for the company.

9. Break down barriers between departments. People in research, design, sales, and production must work as a team, to foresee problems of production and in use that may be encountered with the product or service.

10. Eliminate slogans, exhortations, and targets for the work force asking for zero defects and new levels of productivity. Such exhortations only create adversarial relationships, as the bulk of the causes of low quality and low productivity belong to the system and thus lie beyond the power of the work force.

11. a. Eliminate work standards (quotas) on the factory floor. Substitute Leadership.

    b. Eliminate management by objective. Eliminate management by numbers, numerical goals. Substitute leadership.

12. a. Remove barriers that rob the hourly worker of his right to pride of workmanship. The responsibility of supervisors must be changed from sheer numbers to quality.

b. Remove barriers that rob people in management and in engineering of the right of pride of workmanship. This means, *inter alia,* abolishment of the annual or merit rating and of management by objective.

13. Institute a vigorous program of education and self- improvement.

14. Put everyone in the company to work to accomplish the transformation. The transformation is everybody's job. [Ref. 20]

Actual improvements can be achieved by the Plan-Do-Check-Act (PDCA) cycle discussed by Deming [Ref. 20]. This cycle provides management with a mechanism to make and observe changes placed on a process. It is a continuous cycle. The first step is to plan for a change. This plan should then be carried out or tested (the do), preferably on a small scale. The third step (check) is an important part of this cycle. It consists of observing the effects of the change. The final step, means to take appropriate action from what was learned.

Observations of the PDCA cycle will give management greater knowledge as the cycles are repeated, changed, and implemented. This is a continuous process, stimulated by feedback. Deming also speaks of profound knowledge. One can not improve a process unless one has profound knowledge of the process. The 14 points, PDCA cycle, and the system of profound knowledge together form the framework for Deming's TQM. Profound knowledge should include systems theory, statistical theory, the psychology of people, as well as a deep understanding of learning and change. The result is quality production.

The next section describes how TQM principles can be applied to the software acquisition process.

C. SOFTWARE TOTAL QUALITY MANAGEMENT

In this section possible applications of TQM to the process of software acquisition are discussed. For the purpose of this thesis, quality software is defined as the absence of errors or defects from the end product. Some would argue that totally error free software is almost impossible to achieve. Yet, in applying the provisions of TQM, error free software is the ultimate goal. Continuous improvement in the software development process, can approach the goal of error free software. The remainder of this chapter will focus on software TQM.

Richard E. Zultner [Refs. 21, 22] provides some insights and potential examples for applying the TQM methodology to software development. He first modifies Deming's 14 points. For example; point 5 describes the importance of process control, point 6 emphasizes the essentiality of education, and point 9 shows that breaking down barriers in communication is necessary to produce quality software. The entire set of adopted points are listed in Appendix D. He also changes Demings' seven deadly diseases for software quality. The following seven areas should be avoided to attain quality software:

1. Lack of constancy of purpose to develop software that will satisfy users, keep software developers in demand, and provide jobs.

2. Emphasis on short-term schedules--short-term thinking (just the opposite of constancy of purpose toward improvement), fed by fear of cancellations and layoffs, kills quality.

3. Evaluation of performance, merit rating, and annual reviews--the effects of which are devastating on individuals, and therefore quality.

4. Mobility of software professionals and managers. Job hopping makes constancy of purpose, and building organizational knowledge, very difficult.

5. Managing by "visible figures" alone--with little consideration of the figures that are unknown and unknowable.

6. Excessive personnel costs. Due to inefficient development procedures, stressful environment, and high turnover, software development person-hours are too high.

7. Excessive maintenance costs. Due to bad design, error ridden development, and poor maintenance practices, the total lifetime cost of software is enormous.

Zultner also evolves these principles into areas that should be avoided while designing quality software. For example, when the user's requirements are continuously changing the result will be poor quality software. He concludes with an excellent synopsis: "When you build better, you do each task only once--no rework, no fixes necessary--and that's the fastest (and cheapest) way to build software." [Ref. 22]

In an article in *Program Manager,* Lieutenant Colonel Anthony F. Shumskas, USAF, provides some excellent examples on how to apply TQM to software acquisition [Ref. 23]. He indicates that DOD-STD-2167A (Defense System Software Development) and DOD-STD-2168 (Defense System Software Quality Program) provide the structure for applying TQM to the process of software development. This reduces errors and results in higher quality software. In general terms, Shumskas states that managers' commitment to the user's needs (as specified in the requirements definition), continuous improvement, and education are essential elements for quality software acquisition. He identifies three unique attributes to software TQM as being: process control (process maturity model); standardization (Ada); and metrics (measuring devices). A detailed discussion on these three items is presented in sections D, E, and F. Lieutenant Colonel Shumskas' evaluation indicates that applying TQM to software acquisition has resulted in a decrease of the following: 35 percent savings in development cost, a 50 percent decrease in test time and cost, and 50 percent decrease in documentation costs [Ref. 23].

Captain Roj Karimi, USAF, reinforces the notion that TQM can be applied to the process of software acquisition [Ref. 24]. His main point is that control of the software process may be measured by using a basic set of metrics to measure schedule, effort, code production and test. Applying statistical control to these metrics to determine process deviation results in higher quality software. He summarizes by stating that the goal is to obtain 100 percent quality software for the customer.

It is important to realize that the TQM philosophy can be applied to software acquisition. Basic TQM principles such as customer satisfaction, education, elimination of barriers to communication, and management's commitment to change are essential elements to any software TQM initiatives. In addition there are three items that specifically apply to software TQM: process control, standardization, and metrics.

D. PROCESS CONTROL

In 1984, the United States Air Force, Electronics System Division, Hanscom AFB, contracted the Software Engineering Institute (SEI), located at Carnegie-Mellon University, to investigate and analyze the current software acquisition process. Their first step was to define the software process. SEI determined the following:

> The software process is the set of activities, methods, and practices which guide people (with their software tools) in the production of software. An effective process must consider the relationships of the required tasks, the tools and methods, and the skills, training, and motivation of the people involved. [Ref. 25]

This definition provides the basic foundation for the software acquisition process.

As a result of this investigation, SEI constructed a Software Process Maturity Model, Figure 3-1 [Ref. 25:Fig. 1.2.1], as a tool to improve the software acquisition process. There are five levels. The maturity levels are defined as follows:

1. Initial Level (1). This level has no defined process and is chaotic. It lacks formality in plans, procedures, integration, and management commitment for improvement. This level is devoid of virtually all TQM principles.

2. Repeatable Level (2). Management begins to get involved to establish only basic controls. The software process is not yet clearly defined. This level starts to employ some of the TQM beliefs.

3. Defined Level (3). This level introduces software engineering principles, architecture, and technologies. The organization has now established a process with the realization that the process will require continuous improvement.

4. Managed Level (4). Data are now gathered and analyzed, yet the full utility of this information is not readily apparent. A minimum number of quantitative parameters for quality control and productivity are established.

5. Optimizing Level (5). Data collection is automated, and defect prevention is used. Customer satisfaction is the number one priority. This level is marked by an organization that has attained control over their process; they focus on continuous improvement. This is the ultimate goal of software TQM.

Examination of the model leads one to realize that as the maturity level increases from the initial level to subsequently higher levels, the risk of producing poor quality software is diminished.

TABLE 3-1. SEI SOFTWARE PROCESS MATURITY MODEL

Level	Characteristic	Key Problem Areas	Result
Optimizing	**Improvement fed back into process**	**Automation**	**Productivity & Quality**
Managed	(quantitative) Measured process	Changing technology Problem analysis Problem prevention	
Defined	(qualitative) Process defined and institutionalized	Process measurement Process analysis Quantitative quality plans	
Repeatable	(intuitive) Process dependent on individuals	Training Technical practices ● reviews, testing Process focus ● standards, process groups	
Initial	(ad hoc/chaotic)	Project management Project planning Configuration management Software quality assurance	**Risk**

As previously outlined, one of the key elements of TQM is to define the process and then continuously improve it to obtain quality products. The SEI software process maturity model provides an organized approach for implementing software TQM. As the process is improved by moving from one level to the next higher level the software quality is also increased. The SEI model provides guidelines for management to use in determining their maturity level. Management's goal is to be at the optimizing level.

The SEI has made an assessment of various software producing organizations. A questionnaire, about 120 questions, and an onsight verification are used to determine the appropriate maturity level of an organization. As of April 1991, with 296 projects reviewed, SEI had determined the following: 88 percent of the software projects were at Level (1), five percent were at Level (2), five percent were at Level (3), no projects were found to be at Level (4), and two percent were at Level (5) [Ref. 26]. This means that software TQM principles are not currently employed in the software industry. The SEI Software Process Maturity Model is an excellent measuring device that management can use in determining where they currently are with regards to good software development techniques.

The DOD TQM Maturity Model, Appendix E [Ref. 27], is an adaptation of SEI's Software Process Maturity Model to incorporate software TQM issues. This model identifies eight TQM categories. It relates them to SEI's maturity level by stating the traits associated at a given level. For example, in a Level (1) organization, software tools and statistics on the process are not identified or used.

As an organization improves its process, tools begin to be used. The highest level is marked, not only by the use of tools, but also by statistical process control being known and applied by all employees. This model illustrates how TQM can be directly correlated to the SEI maturity model.

The bottom line is that management's goal is to be at maturity Level (5), the point where good TQM philosophies are incorporated in the design and development of software. The next section will discuss the principle of standardization and how it applies to software acquisition.

E. STANDARDIZATION

An essential element for the implementation of a repeatable and continuously improving process is standardization. This important element is directly applicable to a process of software acquisition that embraces the TQM philosophy of reducing variability. DOD Instruction 5000.2, Part 6, Section D, specifies Ada as "...the only programming language to be used in new defense systems and major software upgrades of existing systems...when cost effective" [Ref. 9].

Ada is a DOD trademark. As such a compiler can only be termed an "Ada compiler" after having passed a series of tests/benchmarks which certify it compiles with the ISO standard for the Ada higher order language (HOL). Only then will the DOD approve the use of the term "Ada." By using the Ada language for weapon systems software, DOD achieves a significant degree of standardization.

With standardization comes the feasibility of more effective software reuse. Different variants of languages and compatibility problems can be eliminated. Software can be designed and developed for specific packages. These packages can be integrated to form a system. Developing modular packages for specific functions leads itself to the concept of software reuse for different applications. Reusable software: "[The] key breakthrough in object technology is the ability to build large programs from lots of small, prefabricated ones [Ref. 28]."

Standardization provides the potential to reduce the process variation presently facing software acquisition, an essential philosophy of TQM. It also should ease the portability of software to allow for use across different computers.

F. METRICS

Software metrics are tools to be used by managers to assist them in their ability to measure the progress of software development. Statistical process control (SPC), an essential element of TQM, is the ability to quantitatively measure any production process to determine common or special causes of variation that produce poor quality products. Software metrics provides these measuring devices. This section reviews the set of metrics currently used by the MICOM Software Engineering Directorate (SED).

The MICOM SED was established in July 1984 with the requirement to provide the Mission Critical Computer Resource (MCCR) expertise needed for weapon systems over their life cycle. As a management tool SED constructed a standard set of metrics for the MICOM community consisting of eight items. This set of metrics is a living document. The metrics are subject to change pending feedback from contractors, program offices, and field requirements. Presently the SED is measuring eight items. The following is a brief summary [Ref. 29]:

1. Requirements Stability: measures the number of software requirements that change from the given baseline. The first step in software development is to identify each software requirement specification. Each separate requirement generally starts out by stating "the software shall..." After the total number of requirements are defined they are tracked as a function of time. Over time, additional requirements are frequently added. Ratings of red, amber, and green are assessed on the percent deviation from the base software specification.

2. Software Development Manpower: this metric measures the developer's actual effort against the developer's approved plan for effort allocation. The contractor provides an estimated manpower requirement, and this indicator monitors the actual verses the planned personnel requirement. Ratings of red, amber, and green are given based on the actual deviation from the original manpower requirement baseline.

3. Software Development Progress: measures the completeness of the software development. The developer provides a developmental plan across all phases and work elements of the software development. This indicator measures the actual to planned progress of the development for the software. The red, green, and amber assessments are based on percentage deviation from the actual schedule.

4. Computer Resource Utilization: measures the utilization of each computer processor, memory, and input/output channel. This is a straight forward indicator that measures the percent utilization for the above items. A significant reserve capacity is desired to allow for future growth capacity.

5. Schedule Risk Analysis: measures the developer's schedule against their actual schedule. The primary purpose is to assess the realism of the developer's schedule. This measurement compares the development schedule to the cumulative man months expended.

6. Trouble Report Resolution: measures the relative number of trouble reports within a specified time period. These reports are categorized as catastrophic, severe, or moderate. The red, amber, and green assessments are based on the number and severity of the trouble reports.

7. Software Product Delivery: measures the timeliness and acceptability of the software products. This is a straightforward measurement of planned to actual product deliveries. Red, amber, and green assessments are based on acceptable delivery.

8. Software Supportability: Uses a 16 factor model to determine Post Deployment Software Support (PDSS) supportability. Such factors as documentation, testing, and supportability are assessed. Red, amber, and green ratings are determined by the summation of these factors.

These metrics were constructed to have a minimal cost impact on the software development process. An excellent source of other examples of metrics can be found in the Air Force Systems Command, Electronics Systems Division and MITRE document, ESD-TR-88 [Ref. 30].

Metrics play a critical role in the TQM of software acquisition by providing quantitative means of gathering data. "Management metrics let you assess both the process and the product at periodic data points, predict trends in the process and product parameters, and specify changes to be made to the process in response to identified problems so you can influence the product as it is being developed." [Ref. 31]

G. SUMMARY

This chapter presented the concept of TQM as a focus on the entire software acquisition process. Conventional software acquisition focuses on the resultant software quality at the end of the "black art" process. The ultimate customer needs are the essential element for the proper implementation of TQM. Commitment from upper management, as shown by General Tuttle's emphasis on the importance of software, is another TQM philosophy. SEI has designed a model to define the software acquisition process at various maturity levels; it is a measurement device to be used to constantly improve the process. Standardization and the use of Ada will reduce variability in the process. Metrics provide a measuring technique to assess the software acquisition process. Process control, the SEI model; standardization, Ada; and metrics, for SPC, are unique Software TQM traits. Software TQM defines a process and continuously improves it. Adhering to software TQM will enable producers to approach Level (5). This is the ideal approach to software acquisition. This chapter has demonstrated that TQM is applicable to the process of software acquisition within DOD.

IV. CASE STUDY: ARMY TACTICAL MISSILE SYSTEM

A. INTRODUCTION

The Army Tactical Missile System (Army TACMS) and the Multiple Launch Rocket System (MLRS) were commended for attaining exemplary performance during the Gulf War. Stephen K. Conver, the Army Acquisition Executive, stated: "...that Army TACMS destroyed, or rendered inoperable, all of its targets [Ref. 6]." He also indicated that MLRS's "rain of steel," fired more than 10,000 rockets attaining outstanding performance. The Army Times indicates the proposed 1993 defense budget will include both Army TACMS and MLRS, the "[stars] of the Persian Gulf war [Ref. 32]." Both of these systems are software intensive, and have proven themselves in a combat environment.

The MLRS is the firing platform for Army TACMS. The software interfaces between the two systems have posed a managerial challenge. Subsequent sections in this chapter will present the evolution of Army TACMS. The first section introduces and explains the basic MLRS as it applies to Army TACMS. The primary focus is on software and software related issues concerning Army TACMS. The final section presents possible future applications for the Army TACMS.

B. MULTIPLE LAUNCH ROCKET SYSTEM

The MLRS is a derivative of the Bradley Fighting Vehicle, consisting of a mounted rocket launcher that is an all weather, highly mobile, multiple launch rocket system. The MLRS mission requires rapid response with high volume, indirect fire with conventional munitions against enemy artillery units, air defense batteries, and lightly armored equipment and personnel. This system is fielded to U.S. Army divisional units as batteries (9 launchers), and to corps units as battalions (3 batteries). This Field Artillery (FA) system is employed in a general support artillery role. [Ref. 33]

The basic MLRS production contract was awarded in 1977 with an Initial Operational Capability (IOC) in March 1983. The contractor structured the software architecture with low risk technology. As a result there was virtually no room for software expansion, without expensive hardware retrofits. This computer technology, from the mid 1970's, rapidly eroded into obsolescence.

The Fire Control System (FCS) is software intensive. The FCS contains the software that is responsible for ballistic computation, navigation, reloading, display, command, control, communication, and automated aiming for MLRS. The primary FCS components are:

- Fire Control Panel (FCP): The FCP is the system input/output device where the operator interfaces with the FCS.

- Electronics Unit (EU): The EU is the computer and memory that calculates the necessary data for the fire missions of the system.

- Stabilization Reference Package/Position Determining System (SRP/PDS): The SRP/PDS establishes the elevation, azimuth, location, and position of the system as it moves from point to point, on a continuous basis.

- Fire Control Unit (FCU): The FCU provides the electronic coordination among the FCS, SRP/PDS, and the electromechanical interfaces of the system.

The original FCS hardware, contained in the EU, was program read only memory (PROM). This antiquated system required a hardware modification (removal and replacement of the PROM) whenever the software requirements changed. Upgrading the PROM software required inserting new boards into the EU. This required the retrofitting of all fielded units, a costly endeavor. In addition, the EU software memory requirements soon expanded, using up the remaining memory capacity. This greatly restricted the possibility for future growth or change.

Also, the software architecture for the original FCS did not support the addition of new munitions or provide for growth of the existing set of munitions that were fired from the basic MLRS. Rapid expansion of software requirements and future requirements necessitated the need to improve the existing computer hardware for the MLRS system. The following section presents the software improvements and the changes to the MLRS hardware architecture, to meet the requirements for Army TACMS.

C. ARMY TACTICAL MISSILE SYSTEM

The Army's need for the Army Tactical Missile System was first established in a mission need statement (MNS) which outlined the need for a mobile, all weather, rapid response, deep attack conventional missile system that could attack high value corps targets at ranges beyond the capability of existing cannons and rockets [Ref. 34]. The thrust of the Army TACMS requirement was to integrate an inertially guided missile into the already existing MLRS that would dispense a payload of M74 submunitions on target locations. In addition, Army TACMS was designed to integrate with the previously established logistic support structure for MLRS. This included existing material handling equipment and transportation modes. The deployment of Army TACMS was to require no increase in force structure. Personnel requirements were to come from the compression of the existing Corps LANCE units into nuclear only battalions and use of existing force structure available. This system was designed to augment the current Field Artillery (FA) capabilities. [Ref. 35]

In 1978, LTV was awarded a demonstration contract to determine the feasibility for a long range system to attack enemy second echelon forces beyond the capability of existing cannons and rockets. This produced a software algorithm, termed the waypoint guidance method. This software required a missile to fly to predetermined points in space as the most risk free way to increase the accuracy and probability of hitting a desired target.

The initial production contract for Army TACMS was awarded in March 1986 to LTV Aerospace and Defense, Dallas, Texas. This was the same contractor responsible for the development and support of the MLRS. The contract type was awarded as fixed price incentive firm (FPIF) [Ref. 36:1.2.4.1c]. Largely, as a result of the contract type, coupled with the software technology already in the contractor's possession, the contractor delivered software which met the minimum Army TACMS requirements.

In addition to the initial contract, an integration contract was awarded of Army TACMS into MLRS sole source to LTV. This required an upgrade to the existing MLRS hardware to accommodate Army TACMS software requirements. The MLRS software was to be rehosted into new computer hardware. Initially there was a problem of having two different project offices in charge of the software acquisition and integration requirement for a combined system. As a result, the Program Executive Officer (PEO) assigned the responsibility for software configuration management and release authority to the MLRS project office. Subsequent to the PEO's decision, the MLRS project office has maintained the software for the entire MLRS family of munitions (MFOM) to include Army TACMS. The MLRS project recently established an MFOM interoperability testbed at SED. It provides a means to test software changes, modifications, and upgrades.

To upgrade the MLRS for Army TACMS the FCS required modification to some of the primary components and the addition of other components. This included:

- Improved Electronics Unit (IEU): The IEU has three microprocessors and the ability to be reprogrammed in the field by use of a Program Load Unit (PLU). With this hardware modification software upgrades can now be accomplished with the PLU. No physical modifications (i.e. PROM) are required. In addition, the memory capacity was increased to provide for potential growth.

- Improved Stabilization Reference Package/Position Determining System (ISRP/PDS): The ISRP/PDS provides the same functions as the SRP/PDS. It also provides greater accuracy and greater speed required to support the guided missiles fired from the MLRS.
- Payload Interface Module (PIM): The PIM provides the interface among the launcher, IEU, and the payload (missile).

This launcher configuration, employing the IEU, ISRP/PDS, PIM, and the current software, is referred to as the Deep Attack Launcher (Army TACMS). To provide more growth potential and avoid obsolescence, a new FCS is being developed for fielding in the 1998 time period.

Although the software changes themselves may be difficult, software changes to the Army TACMS can now be easily accommodated as a result of these hardware modifications to the basic MLRS launcher. Software changes resulting from operational testing and field requirements have been incorporated into the system, after verification, via the PLU. They required no hardware changes. The ability to enhance Army TACMS through software changes has grown substantially as a result of the increased hardware capacity.

The Army TACMS software, approximately 200,000 lines of code for the MFOM, did attain the goal of being on schedule and within budget. There have been a few problem areas encountered, as with all software intensive programs, for this missile system. For the eight software management indicators, as compiled by SED, the Army TACMS scored 5 green, 1 amber, and 2 red. The amber score in trouble report resolution was due to the software changes made to incorporate the explicit guidance. The first red rating in computer resource utilization was due to memory reserves exceeding 50 percent capacity. The other red rating in supportability was due to a lack of documentation. All other areas were rated green. A more detailed discussion on these anomalies will be presented in the analysis chapter.

The next section presents possible future developments and enhancements in software capabilities that the Army TACMS project office is evaluating. These programs are all software intensive.

D. FUTURE DEVELOPMENTS

The Army TACMS project office is presently assessing three new program options: the Block II, extended range, and the enhancement program. All represent increased combat potential for the Army TACMS missile. These systems will require software developmental efforts that will use the existing hardware on the MLRS platform.

The Block II program is designed to incorporate thirteen smart submunitions into a modified Army TACMS missile. These smart munitions, referred to as Brilliant Anti-Armor Submunitions (BATs), will have the mission to kill tanks, armored personnel carriers, self-propelled vehicles, trucks, and personnel. The Block II missile will fly to a predetermined dispense point and then release the BATs which will then search for their targets. The software for the Block II will be written in Ada, and will interface with the existing MLRS software (JOVIAL). There will be no hardware change to MLRS, the only change will be the addition of application specific programs for this new type of munition.

The extended range missile decreases the payload of the conventional M74 submunitions by over two thirds and adds a Global Positioning System (GPS) to provide greater navigational accuracy. By extending the length of the rocket motor by eighteen inches, other potential enhancements are possible. With the additional range and the GPS, a new payload of (six BATs, high explosive, or conventional M74 submunitions) could be used to destroy enemy targets at increased distances with greater accuracy. A software change will be required to program this new configuration.

Software, software development, and software support will play a major role in the future programs associated with Army TACMS. Army TACMS has clearly demonstrated growth potential for new missions and emerging munitions.

E. SUMMARY

Army TACMS and MLRS, both software intensive systems, have proven themselves in a combat environment. This chapter was devoted to the historical development that has lead to their success. The following chapter will conduct an analysis with respect to possible TQM applications to the software.

V. ANALYSIS WITH LESSONS LEARNED

A. TQM APPLICATIONS

This section presents the case analysis for the Army TACMS. Special emphasis will be placed on software acquisition areas and the potential use of software TQM principles that could be used during the acquisition process. The first section presents general TQM applications to the process of software acquisition. Subsequent sections focus on the three primary software TQM principles of process control, standardization, and metrics, as they apply to Army TACMS.

1. Software TQM

Basic TQM beliefs are directly related to good management and production of quality products during the acquisition process. Deming's and Zultner's principles and philosophies that apply to software related issues and the process of TQM for Army TACMS are discussed in the following paragraphs.

Two of the critical aspects of TQM are management's commitment to produce quality products (Deming's point 7) and to satisfy the customer (Deming's point 1). All personnel interviewed in the Army TACMS program office, the MLRS program office, SED, and the contractor's office have shown commitment to the principle to produce quality software products. For example, they all use feedback from the customer (user) to modify the software products to create a more friendly interface between the user and the application software. The bottom line is that upper management, commensurate with GEN Tuttle's beliefs, is committed to the customer's needs.

Another important aspect of TQM is the institution of a software education and a software self improvement program (Deming's points 6 and 13). Although TQM principles are being taught in the MICOM community, there is no structured educational program for software specific requirements. In addition, elimination of barriers to communication (Deming's points 9 and 12) is essential for software quality. There also appears to be some animosity among the program offices, SED, and the contractors. These problems stem from the consternation over who supports whom, and who has the ultimate responsibility for PDSS. This results in barriers to communication.

Mobility of software specialists, especially contractor personnel, is counter productive and is frequently cited as a reason for poor quality software products. Zultner's adaptation of Deming's seven deadly diseases (number 4) states that avoiding turnover in software personnel is important in producing quality software products. This issue has confronted the Army TACMS contractor. Presently LTV Corporation is in Chapter 11. This has resulted in a loss of a significant number of software personnel to different organizations that show future potential for sustained long term employment. Personnel continuity is an essential element for the production of quality software products. The DOD PMs should emphasize this with their contractor counterparts.

Basic TQM principles are important in the production of quality software products. This section identifies the basic TQM philosophies for the acquisition of any product. These basic principles are then adapted to the process of software acquisition as they apply to the Army TACMS. The following sections present the specific software TQM principles of process control, standardization, and metrics as they apply to Army TACMS.

2. Process Control

Software TQM applies to the entire software acquisition process. The SEI Software Process Maturity Model provides a road map to software TQM. This section presents the challenges that SED, the contractors, and program offices encounter with respect to defining the various processes and maturity levels these organizations face.

Process definition is a requirement for TQM. All the customers to each portion of the process need to be identified. Proper identification of the entire process is essential. The relationships among the program offices, SED, the users, and the contractors need to be clearly defined.

For software acquisition the program office contracts for the required software products. Then the contractor produces to the requirement. The SED should provide support and expertise to the program office (the customer to SED). This increases the probability the Government will obtain quality software products from the contractor. Once the program office has accepted delivery of the software it is its responsibility to provide service to the customer. This is the first step in contracting for new software.

For PDSS issues, the role of customer and producer are reversed. If SED is held responsible for the software support, they become the customer to the program office. The program office now would have the responsibility to provide a supportable product that meets the customer's needs to the SED for purpose of software sustainment.

Establishing a clearly defined process among the program office, SED, and the contractor is necessary for process definition, a requirement for software TQM. Also, each organization needs to be responsive and understand the needs of the next customer in their chain. In the case of Army TACMS a clearly defined process for the acquisition and support of the software .was not explicitly apparent during the investigatory work for this thesis.

In addition, SED personnel acknowledge that their organization is at the same maturity level as most other software companies, level 2. The SED faces the same problem industry faces of the limited number of available software personnel verses requirements for these software experts. In addition, SED is confronted with the fact that the DOD budget is shrinking. This limits their ability to hire an adequate software engineering staff. At the same time they are experiencing an exponential growth in the software requirements for weapon systems. To effectively increase their maturity level SED needs more experienced personnel. Educational opportunities are also limited by the present budget.

Proper identification of the software acquisition process and the opportunity to increase one's SEI maturity level are essential elements for successful software TQM. As outlined by Deming (points 3 and 5), once the process is identified continuous improvement to that process is required to attain quality products.

3. Standardization

Applying TQM principles to software acquisition requires the use of standardization. Standardization provides a reduction in the variability of the software development process. Presently, both Army TACMS and MLRS software is written in JOVIAL and assembly language. The Block II Army TACMS software, if approved, will be written in Ada. This will present an ever greater standardization problem because there will then be three different languages in use. The lack of standardization in Army TACMS will be alleviated when the new FCS is fielded. The FCS software is required to be written in Ada. Ada will provide standardization in the MLRS family of munitions. At this point one of the key software TQM principles will be realized.

4. Metrics

The set of eight metrics established by the MICOM SED are an excellent starting point for implementing the provisions of software TQM to missile systems. The SED personnel indicated that these metrics will be continuously updated based upon constant feedback on the system. This is a TQM philosophy. The set of metrics provides the feasibility of using SPC to aid in continuous process improvements for software maintenance.

The SED rates Army TACMS as a low risk (green status) in the following areas: requirements stability, software development manpower, software development process, schedule risk analysis, and software product delivery. In terms of software TQM these areas are within statistical control parameters.

The SED rates one Army TACMS software metric as medium risk or amber. This rating was given to the trouble report resolution metric. It is the result of the incorporation of a new flight algorithm, termed explicit guidance, being incorporated into the software. This indicator appears to be a good metric when it is applied to trouble reports that indicate a software problem. The metric indicates the relative frequency, severity, and age of the trouble report. Explicit guidance was actually a software enhancement change that increased the capabilities of the Army TACMS. This indicator appears as amber as a result of the age of the trouble reports associated with the explicit guidance. These trouble reports will be closed when the next version is released. A green rating is then anticipated.

The first red or high risk rating was in the area of computer resource utilization. Army TACMS exceeded the 50 percent utilization requirement for certain memory and throughput reserve capacities. This situation is expected to be remedied with the fielding of the new FCS. A second red rating was given in the area of supportability. The supportability issues are focused on PDSS documentation problems. The lack of documentation and software errors are the major contributors to this rating. This metric is expected to improve as the contractor provides more documentation and resolves software errors.

The SED metrics are living documents that were adopted in June, 1991. These metrics will require modification and further tuning as feedback information is reviewed. They are an excellent start of applying another TQM principle to software acquisition.

B. LESSONS LEARNED

This section presents software lessons learned during this study. First, generic lessons learned are presented. Then lessons learned from the Army TACMS software point of view are given.

1. Generic Software Lessons Learned

General Tuttle has emphasized the importance of software in modern weapon systems. He also describes four generic software lessons learned that the software acquisition community should keep in mind. They are:

- It's much cheaper to introduce changes early [in the process].
- [We] need software models that promote prototyping, iteration and parallelism.
- There's a cost in supporting fielded software that we must incur, even if we don't make changes.
- We must adhere to the DOD's single standard programming language: Ada. [Ref. 1]

2. Army TACMS Software Lessons Learned

The following lessons learned are compiled from the Army TACMS project. Each lesson learned is presented with possible TQM applications.

The most impressive lesson learned is associated with the flight path algorithm. The prime contractor chose a way point guidance flight path due to their experience during the demonstration contract. The way point guidance method was used to minimize the contractor's risk, an important consideration for FPIF contracts.

It was found that the way point guidance raised a survivabiiity issue at long ranges. As a result, a new software algorithm was designed, termed explicit guidance. This algorithm increased the velocity and apogee of the missile to enhance the survivability. As an added benefit the over all range was increased by an incredible 24 percent. Most impressive is the fact that this improvement was accomplished with only a software change. No hardware change was required. Applying software TQM, avoidance of this scenario may be is illustrated by one of Zultner's deadly diseases (number 2): avoiding short term goals. The lesson learned: structure a contract for software requirements such that it is adaptable for future expandability requirements or provides incentives to the contractor to achieve capabilities in excess of the minimum requirements but within budget and schedule. Do not contract for minimum software thresholds.

A second lesson learned is associated with the Army TACMS and MLRS software integration contract. Both project offices were in charge of the software development for their particular applications. The problem encountered was that there was no one organization in charge of the overall development. There was confusion as to which organization had the authority and responsibility to make changes and provide configuration control for the software. This problem was remedied by placing the requirement for software configuration management and support with the MLRS project office for the MFOM. The turbulence was eliminated. The lesson learned: have one organization in charge of overall software related integration, configuration management, and support requirements.

User interface requirements generate another lesson learned. The ISRP requires approximately eleven minutes to load the software for Army TACMS. If the system is shutdown while the ISRP is loading the firmware in the ISRP is damaged. This requires a contractor to perform costly and time consuming hardware maintenance to change out the firmware. The prompt on the FCP that warns of this problem is not highly visible during the ISRP software loading process. As a result ISRP's have been damaged. Avoiding Zultner's first of seven deadly diseases, satisfying user requirements, is the software TQM principle that was over looked in this case. This was because the user's requirements were not clearly identified. The lesson learned: conduct manprint studies to ensure the input/output devices are user friendly.

Another source of problems is encountered by the requirement for SED to yearly contract for support. SED does not currently have the inhouse capability to perform all the PDSS requirements, nor the manpower allocations. Contracted personnel and turn over cause a loss in personnel continuity for the software support process. SED is attempting to alleviate this situation by getting a multi-year contract. However, the potential for loss of continuity still exists. From Zultner's seven deadly diseases, number 4 establishes the need to maintain continuity of personnel for software TQM. The lesson learned: if possible, increase SED's inhouse capabilities to better preserve manpower continuity in the software acquisition process for PDSS. It is felt that there is less turnover of Government personnel.

The quality of contractor personnel presents one more lesson learned. Currently LTV is in Chapter 11, which has caused a large number of software personnel to leave LTV. It has made it difficult for the company to hire software people from the limited pool of available professionals. This loss of continuity causes inefficiency in the software acquisition process (Zultner's deadly disease number 4). The lesson learned: if possible, contract with an organization that is rated above maturity Level (2) and has a proven record for personnel continuity. Level (2) starts to focus on personnel and process groups as an important element in producing quality software.

The last lesson learned pertains to configuration control. During operation Desert Storm, one unit loaded the wrong software revision for the MLRS. This resulted in their inability to fire the conventional MLRS missiles. The lesson learned: configuration control for different software revision levels is essential. Color coding could be a possible solution to this problem. A different color for each revision level would assist the programmer to assure the proper software is installed.

C. SUMMARY

This chapter presented the case analysis for Army TACMS with special emphasis being place on software TQM issues and applications. The importance of life cycle requirements were illustrated. The most significant lessons learned for Army TACMS were presented to provide examples for future software acquisition projects. The end result shows that software TQM philosophies, when properly employed, can result in quality software.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Software acquisition for modern weapon systems presents the biggest challenge for program managers. The requirement for software in these sophisticated systems has grown at an exponential rate over the past four decades. The demand for qualified software engineers far exceeds the available resources of these experts. Most of DOD acquisition top management have a hardware acquisition orientation. Software engineering is considered a "black art" too complex to understand. PDSS constitutes up to 70 percent of the life cycle costs associated with software. Poor software quality is often the result, with cost and schedule over runs common. Such factors are some of the primary contributors leading to software being named the "Achilles' Heel" in weapon systems development. The PM's challenge is to acquire quality software products.

This thesis has found that Total Quality Management can be applied to software acquisition with a resulting improvement in the process. As previously discussed, software TQM focuses on the entire process that produces the software product, not just the final result. The three primary TQM principles most applicable to software acquisition are: process control, standardization, and metrics. Implementation of these principles will help to alleviate the current software crisis.

The SEI has constructed a software maturity model that can be used to evaluate software producing organizations. This model provides a road map to software TQM for organizations wanting to institute software TQM philosophies. Today, most producers are rated at the low end of the scale and have much room for improvement. The SEI model provides an excellent model for software organizations to improve their process and the quality of their software products. The SEI model is dynamic. It has undergone modification to better refine the process. This model will prove its utility as software organizations adopt its philosophies.

Standardization and a set of useful metrics are software TQM principles. Standardization through the use of Ada has the potential to reduce variation and provides the opportunity for software reuse. The set of metrics used by the MICOM SED is an excellent starting point for instituting a standard set of tools to measure the software acquisition process and determine if the process is in SPC. The SED metrics will prove their value as they are refined through the feedback process and continual process improvement.

The Army TACMS is a software intensive system that has proven its effectiveness in operation Desert Storm. The Army TACMS has provided an excellent case study for software acquisition. Not withstanding its success in Desert Storm, important lessons were learned that will prove beneficial to future software acquisition programs. In this work, software TQM principles have been outlined for Army TACMS.

The software TQM philosophy can be used for software acquisition. For this to happen, top management, as evidenced by GEN Tuttle, must be committed to the process.

Software acquisition is the long pole in the tent for systems acquisition. At the same time software is a force multiplier. Commitment to quality software products for the Armed Forces is a mandatory requirement for PMs. The following section draws together five recommendations resulting from this research.

B. RECOMMENDATIONS

1. Implement Software TQM

All software developing organizations should adopt the philosophies of software TQM as outlined in this thesis. Education is an essential element of TQM. When greater software development knowledge is acquired by management, then the probability of producing quality software will also be increased. Adaptation of the SEI maturity model, Ada, and a set of metrics are essential for software TQM. Management needs to focus on the entire process of software acquisition in order to obtain quality software products.

2. Assess the SED Metrics and Seek New Metrics

The set of eight metrics used by SED to provide management information on software development was established in June 1991. This set of metrics, as indicated by SED personnel, are a living document that will be modified as knowledge is gained on their utility. The validity of these metrics will be demonstrated if they indicate a software development project is successful from its start to finish and, when fielded on time and within cost, proves to be successful to the user. In other words, if all the indicators are green for the software development of a project, and this project is successful, then it could be concluded the metrics are a good set of software development indicators. These metrics are an excellent start to SPC of the entire software acquisition process. SED should monitor their accuracy and make adjustments when necessary. They should also continue to search for other metrics which would prove useful.

3. Cross Train Software Personnel

Within the MICOM community there needs to be a greater understanding of responsibilities between the project offices and the SED. A cross training program that assigns SED personnel to a program office would be an excellent opportunity for both organizations to gain greater knowledge of the entire process of software acquisition. The opportunity for SED personnel to work in a project office, for a specified time period, should be considered professional development. It should not be deemed a career jeopardizing move. The result will be mutual respect and understanding by both organizations.

In keeping with the TQM philosophy of serving your customer, software acquisition offers some interesting points. For software the project needs the professional advice of the SED. The program office should be looked upon as the customer of SED. SED should seek to constantly improve their product (support) to their customer (the program office).

At the same time the PM should be looking at SED as one of its customers. Of course, one usually looks at the soldier in the field as the ultimate customer. But SED must also be looked upon as a customer of the PM. SED will eventually become the configuration manager of the delivered software. As such SED will be very concerned that the software development process focuses on the future maintainability of the software. The PM should strive to provide a quality product to his customer, SED. This aspect is frequently overlooked in weapon systems acquisition.

4. Fund SED

PDSS costs are estimated to be as high as 70 percent of the software acquisition life cycle cost. PDSS requirements tend to be neglected. The MICOM SED is understaffed to perform these functions and is required to subcontract for the additional support. Continuity is often lost as contracts are awarded yearly. Software and therefore PDSS requirements are increasing. DOD needs to adequately fund all PDSS organizations to ensure software quality is achieved. Software is critical for our weapon systems. Software quality is a combat multiplier.

5. Use the SEI Model

The SEI Software Process Maturity Model should be used as a guide for evaluating software production and support organizations. Software acquisition should be a continuously improving process. The SEI model may serve as a road map to improve software TQM. The SEI model should be used to assess potential software contractors. To ensure quality software products, a contractor should have a rating of two or more to be considered as a viable software producer.

C. ANSWERS TO RESEARCH QUESTIONS

The following answers are the results of the analysis completed on interviews with the personnel shown in Appendix C.

1. How can the software acquisition process be improved?

- Implement the software TQM principles outlined in this thesis.
- Clearly define user requirements prior to software design. This will ensure a stable design.
- Implement training programs (e.g., TQM, Software Engineering, Systems Engineering) to provide education and a better understanding among all people involved in software acquisition.
- Ensure adequate documentation for PDSS.
- Use Case studies and lessons learned on software acquisition as a management tool.
- Involve the user in determining software capabilities.
- Use independent verification and validation for software evaluation.
- Review subcontractor's software policies.
- Ensure configuration control for the field.
- Properly define software test requirements.
- Plan for software P3I.
- Make software user friendly.
- Eliminate hardware mentality for software.

- Use a system engineering approach.

2. What situations, from a historical point of view, have caused the current software crisis?

- Rapid growth of software requirements.
- Little knowledge of historical software information.
- A focus on hardware acquisition mind set.
- Lack of a systems engineering approach.
- Lack of quality software documentation.
- Lack of adequate PDSS.
- No clearly defined software development process.
- No standardization of programming languages.
- Work breakdown structure not used for software.
- Trying to fix hardware shortcomings with software changes.

3. How can Total Quality Management principles be applied to the software acquisition process?

- Management commitment to TQM.
- Introduction of process control (e.g., SEI Model).
- Standardization (e.g., Ada).
- Use of metrics (e.g., SED indicators).

4. What lessons can be derived from the Army Tactical Missile System software acquisition process?

- Do not use a FPIF contract for software development.
- Do not contract for minimum requirements in software. Provide incentives for future expandability needs.
- Software integration contract should be headed by only one organization.
- Define user requirements up front, make software more user friendly. Involve the user.
- Continuity in software support personnel is needed.
- Contractor personnel turbulence/turnover causes inefficiency.
- Configuration control is very important.
- Plan for software P3I.

5. What questions should a PM ask to ensure that the software acquisition process is under control?

- Continuously ask software related questions as shown in Appendix B.
- Become familiar with and use MICOM SED Metrics as a guide.
- Know and use software regulations and standards (DOD- STD-2167A).

6. What are the similarities and the differences between software and hardware acquisition?

- Software has the potential to get better with age. Hardware normally does not.
- Hardware is tangible, software is intangible.
- Software development is in its youth.
- Detailed test plans are often neglected for software.

- Software development is human intensive and time dependent. Expenditure of more money will not speed up the process of software development.

7. What is the quality of a contractor's programming staff?

- Most software developing organizations are between Levels 1 and 2 on the SEI maturity model.
- LTV is in Chapter 11, and quality software engineers are not attracted to this organization.

D. RECOMMENDATIONS FOR FURTHER STUDY

The following are areas that deserve follow up:

1. SEI Software Process Maturity Model

Determine if software producing organizations know and understand the SEI model. Determine if software producing organizations have increased in their maturity level by using the model. Determine if this model has matured as it has changed during its evolution. Determine if this model can be used for contracting purposes. Determine what private industry thinks of the model.

2. Army TACMS Block II Development

Examine the software challenges facing the Block II development. Determine the interface problems with the two languages of Ada and JOVIAL. Determine the software challenges with the communications software interfaces. Determine the software relationship between the BAT program office and the Army

TACMS program office. Determine SED's responsibility and role for this software development effort.

3. Software Lessons Learned from Desert Storm

Document the software lessons learned during Desert Storm. Determine the strong and weak points for software. Determine what software changes were made to weapon systems as a result of operation Desert Storm. Determine how software was a force multiplier. Analyze the lessons learned.

4. Analyze MICOM SED Metrics

Examine the MICOM SED metrics currently used. Determine if this is a good set of indicators. Determine the reasons for constructing these metrics. Determine if these metrics do indeed predict quality software. Determine if contractors have confidence in these metrics. Determine if these eight metrics will evolve as a function of time. Determine if there are other beneficial metrics that can be used. Determine the costs of using such metrics.

5. PM Software Challenges

Document the challenges that all PMs face during the process of software acquisition. Construct a list that has been compiled from many different sources and PMs. Document lessons learned. Determine how to incorporate better software development processes. Determine how PDSS support is used for sustainment of major weapon systems. Determine if PMs understand software metrics. Determine how to better contract for software in a weapon system contract.

The Army TACMS presents an excellent case study in software acquisition and related software challenges faced by PMs in the DOD. This weapon system has a proven combat effectiveness. Army TACMS demonstrated how software can be a force multiplier in Desert Storm. All personnel interviewed are committed to the combat soldier. Implementation of software TQM has been initiated. Further study on the Army TACMS software related issues will prove to be beneficial to future PMs.

APPENDIX A. LIST OF ACRONYMS

AMC	Army Materiel Command
AR	Army Regulation
ARES	Army Executives for Software
BAT	Brilliant Anti-Armor Submunition
DARPA	Defense Advanced Research Projects Agency
DLSIE	Defense Logistics Studies Information Exchange
DOD	Department of Defense
DSMC	Defense Systems Management College
DSSE	Developmental Software Support Environment
EU	Electronics Unit
FA	Field Artillery
FCP	Fire Control Panel
FCS	Fire Control System
FCU	Fire Control Unit
FPIF	Fixed Price Incentive Fee
GAO	Government Accounting Office
GPS	Global Positioning System
HOL	Higher Order Language
ICs	Integrated Circuits

IG	Inspector General
IOC	Initial Operational Capability
LCSSE	Life Cycle Software Support Environment
MCCR	Mission Critical Computer Resources
MFOM	MLRS Family of Munitions
MICOM	Missile Command
MLRS	Multiple Launch Rocket System
MNS	Mission Need Statement
OT&E	Operational Test and Evaluation
P3I	Preplanned Product Improvement
PEO	Program Executive Officer
PDCA	Plan Do Check Act
PDSS	Post Deployment Software Support
PIM	Payload Interface Module
PLU	Program Load Unit
PM	Program/Product Manager
PROM	Program Read Only Memory
SED	Software Engineering Directorate
SEI	Software Engineering Institute
SLOC	Source Line of Code
SPC	Statistical Process Control

SRP/PDS	Stabilization Reference Package/Position Determining System
TACMS	Tactical Missile System
TOW	Tube-Launched, Optically-Tracked, Wire-Guided Weapon System
TQM	Total Quality Management
TRADOC	Training and Doctrine Command
TSM	TRADOC System Manager

APPENDIX B. PROGRAM MANAGER'S QUESTIONS

The following is a list of appropriate questions a PM should ask with regards to software acquisition. The list is provided as a guide, and is not intended to be all inclusive. All questions should be answered in the affirmative.

I. Basic Questions:

1. Are the provisions of software TQM being implemented?
2. Is the contractor's maturity level known?
3. Is the contractor attempting to increase their maturity level?
4. Are software metrics used?
5. Are the metrics a good indicator of your software acquisition process?
6. Is the standardized Ada HOL being used?
7. Are the user need's clearly identified?
8. Are the input/output devices user friendly?
9. Has the documentation been streamlined, yet complete?
10. Will the contractor's documentation support PDSS?
11. Is the computer utilization below 50 percent?
12. Does the contractor have a software training program?

II. Software Development Questions:

1. Is the software test plan complete?
2. Is all of the documentation in DOD-STD-2167A necessary?
3. Can DOD-STD-2167A be tailored for cost savings?
4. Are the test ranges scheduled?
5. Is the software patch free?
6. Are errors corrected at the source code level?
7. Is the software support element clearly defined?
8. Are the software security requirements identified?
9. Have software risk areas been identified?
10. Has the software configuration management and control been defined?
11. Have configuration audits been established?
12. Will an independent verification and validation be conducted?
13. Does the RFP contain software metrics that the contractor will be required to meet?
14. Can this software be reused for different applications?
15. Has integration testing been defined?
16. Is the software easy to change or update?
17. Are the compilers validated?
18. Has the built in test equipment been defined?

APPENDIX C. LIST OF PERSONNEL INTERVIEWED

1. Matthews, David F., COL USA. Program Manager, Army Tactical Missile System, Redstone Arsenal, AL, Interviews, November 1991 and January 1992.

2. Flohr, Steven W., LTC USA. Product Manager, Block II, Army Tactical Missile System, Redstone Arsenal, AL, Interview, January 1992.

3. Maxwell, John M., Assistant Project Manager, Integration, Army Tactical Missile System, Redstone Arsenal, AL, Interviews, November 1991 and January 1992.

4. Evans, Jerry, Chief Computer Resources Branch, Army Tactical Missile System, Redstone Arsenal, AL, Interview, January 1992.

5. Williams, Woodrow A., Computer Resources Branch, Army Tactical Missile System, Redstone Arsenal, AL, Interview, November 1991.

6. Boydstun, Byron R., Electronics Engineer, Multiple Launch Rocket System, Redstone Arsenal, AL, Interview, January 1992.

7. Fitzpatrick, Willie, Software Evaluation, Test and Control Division, MICOM Software Engineering Directorate, Redstone Arsenal AL, Interview, January 1992.

8. Moore, Robert L., Acting Chief, Deployed Systems Division, MICOM Software Engineering Directorate, Redstone Arsenal, AL, Interview, January 1992.

9. Eubanks, James E., P.E. Senior Systems Engineer, Barrios Technology, Huntsville, AL, Interview, January 1992.

10. Quirk, Michael P., Senior Systems Engineer, Barrios, Technology, Huntsville, AL, Interview, November 1991.

APPENDIX D. THE 14 POINTS FOR SOFTWARE MANAGERS [Ref. 21]

1. Create constancy of purpose for the continuous improvement of software and service, with a long term plan to become excellent, satisfy users, and provide jobs.

2. Adopt the new philosophy. We are in a new age of software engineering and software quality. Software managers must awaken to the challenge, learn their responsibilities, and take on the leadership for change.

3. Cease dependence on mass inspection (especially testing) to achieve quality. Reduce the need for appraisal by building quality into the software in the first place. Inspections and testing are not the answer. They are too late and unreliable--they do not produce quality.

4. End the practice of awarding business on price alone. Instead minimize total cost. Move toward a single source for any one item or service, making them a partner in a long-term relationship of loyalty and trust.

5. Constantly and forever improve the software development process, to improve quality and productivity, and thus constantly decrease the time and cost of software. Improving quality is not a one-time effort.

6. Institute training on the job. Everyone must be well trained, as solid skills are essential for improvement.

7. Institute leadership. It is a leader's job to help their people, not judge them. It is to know when their people need special help, and provide it. Supervision of software managers and professionals is in need of an overhaul.

8. Drive out fear. Develop trust so that everyone may work effectively. Management should be held responsible for faults of the organization and environment.

9. Break down barriers between areas. People must work as a team. They must cooperate to foresee and prevent problems during software development and use.

10. Eliminate slogans, exhortations, and targets that ask for zero defects, and new levels of productivity without providing methods. Slogans don't build quality software.

11. Eliminate arbitrary numerical quotas and goals. Substitute leadership. Quotas and goals (such as schedules) address numbers--not quality and methods.

12. Remove barriers to pride of workmanship. The responsibility of project managers must be changed from schedules to quality.

13. Institute a vigorous program of education and encourage self-improvement for everyone. There must be a continuing education and application commitment by managers and professional staff.

14. Put everyone to work to accomplish the transformation. The transformation is everyone's job. Everyone has a part to play in improvement. Management's job is to accomplish the transformation.

APPENDIX E. DOD MATURITY MODEL [Ref. 27]

Maturity Level > TQM Category	1	2	3	4	5
Top Management Commitment	Traditional approach to quality control: Inspection primary tool, better quality = higher cost.	Balance of long-term goals with short-term objectives.	Adequate money and time allocated to continuous improvement and training.	Focus is on improving the system.	Continuous improvement is a natural behavior even during routine tasks.
Obsession with Excellence	Traditional quality control approach.	Corporate council set up.	TQM support system set up and in use.	Use of cross-functional improvement teams.	Constant, relative improvement in quality, cost, and productivity.
Organization is Customer-Satisfaction Driven	Traditional quality control approach.	Customer rating of company is known.	Customer feedback used in decision making	Striving to improve value to customers is a routine behavior.	Customer satisfaction is the primary goal. More customers desire long-term relationship.
Supplier Involvement	Traditional quality control approach.	Suppliers know your TQM direction: supplier number reduction started	Direct Involvement in supplier awareness training: supplier criteria in place.	Suppliers actively implementing TQM philosophy.	Suppliers fully qualified in all benchmark areas.
Continuous Learning	Traditional quality control approach.	Training plan developed	Ongoing training programs.	Top management understands and applies TQM philosophy.	Training in TQM tools common among all employees.
Employee Involvement	Traditional quality control approach.	Manager presents ideas and invites questions, makes decisions.	Manager presents problem, gets suggestions, makes decisions	Manager defines limits, asks group to make decisions.	People involvement, self-directing work groups.
Use of Incentives	Traditional quality control approach.	Effective employee suggestion program used.	Quality related employee selections and promotion criteria.	More team than individual incentives and rewards.	Gainsharing (Cross-functional work groups).
Use of Tools	Traditional quality control approach.	SPC (and other tools) used in manufacturing.	SPC (and other appropriate tools) used for variation reduction.	Appropriate tools used for analysis & improvement of processes.	Statistics is a common language among all employees.

LIST OF REFERENCES

1. Tuttle, Gen. William G.T. Jr., USA, *The Importance of Software,* Briefing to Army Executives for Software, 17 July 1991.

2. Defense Systems Management College, *Mission Critical Computer Resources Management Guide,* 1990.

3. Bohem, Barry, "Software Engineering," *IEE Transactions on Computers,* vol. C-25, no. 12, December 1976.

4. Kitfield, James, "Is Software DoD's Achilles' Heel?" *Military Forum,* July 1989.

5. Bohem, Barry, "High Integrity Systems and Software Risk Management," *Software OT&E Symposium,* 1 November 1991.

6. Conver, Stephen K., Army Acquisition Executive, *Army Research, Development & Acquisition Bulletin,* May-June 1991.

7. Mullen, BG William J. III, USA, *An Army Operational Perspective,* Briefing to Army Executives for Software, 17 July 1991.

8. Department of Defense Directive 5000.1, *Defense Acquisition,* February 23, 1991.

9. Department of Defense Instruction 5000.2, *Defense Acquisition Management Policies and Procedures,* February 23, 1991.

10. Department of Defense Standard, DOD-STD-1467, *Software Support Environment,* 19 January 1985.

11. Department of Defense Standard, DOD-STD-2167A, *Defense System Software Development,* 29 February 1988.

12. Department of Defense Standard, DOD-STD-2168, *Defense System Software Quality Program,* 29 April 1988.

13. Department of Defense Military Handbook, MIL-HDBK-286, *Software Quality Evaluation,* 31 December 1985.

14. Department of Defense Military Handbook, MIL-HDBK-287, *A Tailoring*

Guide for DOD-STD-2167A, Defense Systems Software Development, 11 August 1989.

15. Department of Defense Military Handbook, MIL-HDBK-783, *Software Support Environment Acquisition,* 28 February 1989.

16. AMC Regulation 70-16, *Management of Computer Resources in Battlefield Automated Systems,* 5 April 1990.

17. AMC Pamphlet 70-13, *Software Management Indicators, Insight,* 31 January 1987.

18. AMC Pamphlet 70-14, *Software Management Indicators, Quality Insight,* 30 April 1987.

19. Department of Defense, *Total Quality Management Guide,* Final Draft, 15 February 1990.

20. Deming, W. Edwards, *Out of the Crisis,* Cambridge: MIT Center for Advanced Engineering Study, ISBN 0-911379-01-0, 1986.

21. Zultner, Richard E., *The Deming Way to Software TQM,* Zultner and Company, copyright 1990.

22. Zultner, Richard E., *Software Total Quality Management, What Does It Take to be World-Class?* Zultner and Company, copyright 1990.

23. Shumskas, Lt. Col. Anthony F., USAF, "Applying Total Quality Management to the Software Life Cycle," *Program Manager,* March-April 1991.

24. Karimi, Capt. Roj, USAF, "Total Quality Management in Software Development," *Program Manager,* July-August 1991.

25. Humphrey, Watts S., Kitson, David H., and Kasse, Tim C., *The State of Software Engineering Practice: A Preliminary Report,* Software Engineering Institute Technical Report CMU/SEI-89-TR-1, Carnegie Mellon University, February 1989.

26. Baumbert, John, "New SEI Maturity Model Targets Key Practices," *IEEE Software,* November 1991.

27. Garcia, Suzanne M., *Relationships Between the SEI SW Process Maturity Model, Cho's Statistical Process Control Model, and TQM,* Lockheed Missiles & Space Co. (LMSC), Sunnyvale, California, 1990.

28. Veriety, John W., and Schwartz, Evan I., "Software Made Easy," *Business Week*, September 30, 1991.

29. U.S. Army Missile Command, Research, Development & Engineering (RD&E) Center, Software Engineering Directorate (SED), *Methodology for the Management of Software Acquisition*, June 1991.

30. Schultz, Herman P., *Software Management Metrics,* Air Force Systems Command, Electronics Systems Division, Report ESD-TR-88-001, May 1988.

31. Fenick, Frank, "Implementing Management Metrics: An Army Program," *IEEE Software,* March 1990.

32. Naylor, Sean D., "A Farewell to Arms," *Army Times,* February 10, 1992.

33. Integrated Logistic Support Plan, *Multiple Launch Rocket System*, February 1991.

34. Mission Element Need Statement, *Corps Support Weapon System,* 24 April, 1981.

35. Integrated Logistic Support Plan, *Army Tactical Missile System,* July 1990.

36. Acquisition Plan Number 6, *Army Tactical Missile System,* July 1990.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Defense Logistics Studies Information Exchange U.S. Army Logistics Management Center Fort Lee, Virginia 23801	2
4.	Dr. David V. Lamm, AS/Lt Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	2
5.	Martin J. McCaffrey, AS/Mf Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	2
6.	Dr. Dan Trietsch, AS/Tr Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	1
7.	Colonel David F. Matthews Project Manager Army Tactical Missile System ATTN: SFAE-FS-AT Redstone Arsenal, Alabama 35898-5650	2

8. Willie Fitzpatrick 1
U.S. Army Missile Command
Software Engineering Directorate
ATTN: AMSMI-RD-BA
Redstone Arsenal, Alabama 35898-5260

9. Captain Wayland P. Barber 1
8 Donna Court
Huntington, New York 11743